CLAIMS

What is claimed is:

1. A processor, comprising:

a first exception handler that receives and handles critical excepted instructions; and

a second exception handler that receives and handles non-critical excepted instructions.

2. The processor of claim 1, wherein the critical excepted instructions comprise exceptions that are performance critical.

3. The processor of claim 1, wherein the non-critical excepted instructions comprise exceptions that are not performance critical.

4. The processor of claim 2, wherein the critical excepted instructions include branch mispredictions.

5. The processor of claim 2, wherein the critical excepted instructions include load/store traps.

6. The processor of claim 2, wherein the critical excepted instructions include jump mispredictions.

7. The processor of claim 3, wherein the non-critical excepted instructions include illegal instructions.

8. The processor of claim 3, wherein the non-critical excepted instructions include cache parity errors.

9. The processor of claim 3, wherein the non-critical excepted instructions include invalid instructions.

10. The processor of claim 3, wherein the excepted instructions include arithmetic overflows.

11. The processor of claim 1, wherein the first exception handler operates speculatively.

12. The processor of claim 1, wherein the first exception handler causes critical excepted instructions to be resolved on a speculative basis, even though the excepted instruction may not be in an actual program path.

13. The processor of claim 1, wherein the second exception handler operates non-speculatively.

14. The processor of claim 1, wherein the second exception handler causes non-critical excepted instructions to be resolved only when it is certain that the excepted instruction is in an executing program.

15. The processor of claim 1, further comprising a plurality of pipelines with multiple stages, and wherein excepted instructions may arise in one or more of the pipeline stages.

16. The processor of claim 15, wherein excepted instructions arising from said one or more pipeline stages is routed to the first exception handler or second exception handler based on a predetermined criteria.

17. The processor as in claim 16, wherein the predetermined performance criteria relates to performance of the processor.

18. An exception handler for a processor that resolves excepted instructions , comprising:

a speculative exception handler that receives critical excepted instructions and resolves said critical excepted instructions on a speculative basis; and

a non-speculative exception handler that receives non-critical excepted instructions and resolves said non-critical excepted instructions on a non-speculative basis.

19. The exception handler of claim 18, wherein the speculative exception handler causes critical excepted instructions to be expeditiously resolved even though the critical excepted instruction may not be in an actual path of an executing program.

20. The exception handler of claim 18, wherein the non-speculative exception handler delays resolution of said non-critical excepted instructions until it is certain that said non-critical excepted instruction lies in an actual path of an executing program.

21. A processor, comprising:

at least one pipeline with a plurality of stages;

an algorithm for detecting non-executable instructions in said at least one pipeline, wherein said algorithm generates a command that identifies the non-executable instruction and identifies a reason that the non-executable instruction will not execute;

a speculative exception handler that receives said command for any non-executable instructions that are critical to processor performance; and

a non-speculative exception handler that receives said command for any non-executable instructions that are not critical to processor performance.

22. The processor claim 21, wherein the speculative exception handler expeditiously resolves critical non-executable instructions even though the critical non-executable instruction may not be in an actual path of an executing program.

23. The processor of claim 21, wherein the non-speculative exception handler delays resolution of said non-critical non-executable instructions until it is certain that a non-critical non-executable instruction lies in an actual path of an executing program.

24. The processor as in claim 22, wherein said speculative exception handler includes logic for resolving critical non-executable instructions.

25. The processor as in claim 23, wherein said non-speculative exception handler includes logic for resolving non-critical non-executable instructions.

26. A method of handling exceptions in a processor during the execution of a program, comprising the acts of:

detecting an exception in one or more stages of one or more pipelines;

identifying if the exception is critical to the performance of the processor;

routing critical exceptions to a first exception handler;

routing all non-critical exceptions to a second exception handler; and

expeditiously resolving critical exceptions.

27. The method of claim 26, wherein the critical exceptions are expeditiously resolved by handling the critical exceptions speculatively.

28. The method of claim 26, wherein the critical exceptions are expeditiously resolved even though the critical exception may not be in an actual path of the program.

29. The method of claim 26 wherein a plurality of pipelines are provided for executing multiple program instructions in parallel, and wherein exceptions are detected in each stage of each pipeline.

30. The method of claim 26, wherein the second exception handler delays resolution of said non-critical exceptions until it is certain that a non-critical exception lies in an actual path of the program.

31. A method of handling exceptions in a processor during the execution of a program, comprising the acts of:

detecting an exception and identifying if the exception is critical or non-critical to the processor performance;

routing critical exceptions to a speculative exception handler; and

routing all non-critical exceptions to a non-speculative exception handler.

32. The method of claim 31, wherein the critical exceptions are resolved speculatively, even though the critical exception may not be in an actual path of the program.

33. The method of claim 31, wherein the non-speculative exception handler delays resolution of said non-critical exceptions until it is certain that a non-critical exception lies in an actual path of the program.

34. The method of claim 33, wherein the exceptions from the non-speculative exception handler are selected for resolution prior to the exceptions from the speculative exception handler.